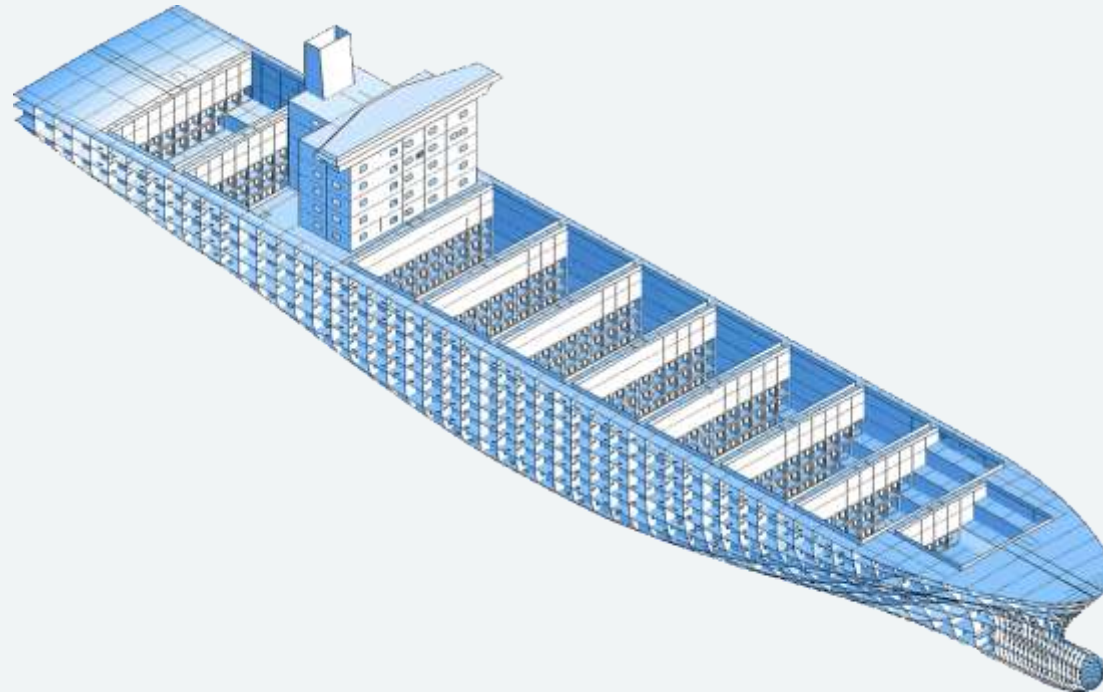


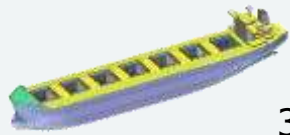


A brief intro to NAPA



NAPA - world's leading software solutions company for ship design and operation

Improve the safety and performance of the global marine industry

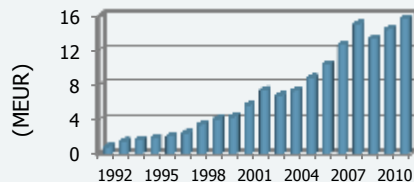


3D modelling



Loading & stability
Energy optimisation
Fleet management

Revenue €16M in 2011
Continuously profitable



Reliability first
Commitment builds trust
Courage leads the way
Enjoy working together
Success through knowledge



Commitment to ecological sustainability and energy efficiency for ships

Shipyards and ship owners
Design companies, institutes and authorities

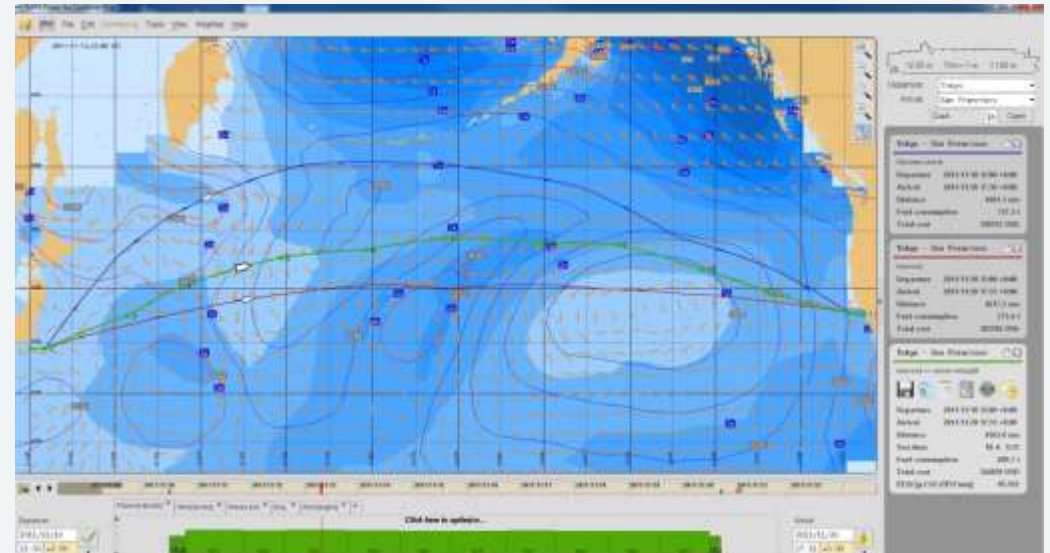
Used by over 700 organizations, installed in 1200 ships

Global reach – Finland, China, Korea, Japan, India, Romania, Italy, USA & worldwide agents

Established in 1989
70% owned by employees
Staff 150, 100 in Helsinki, Finland

NAPA products

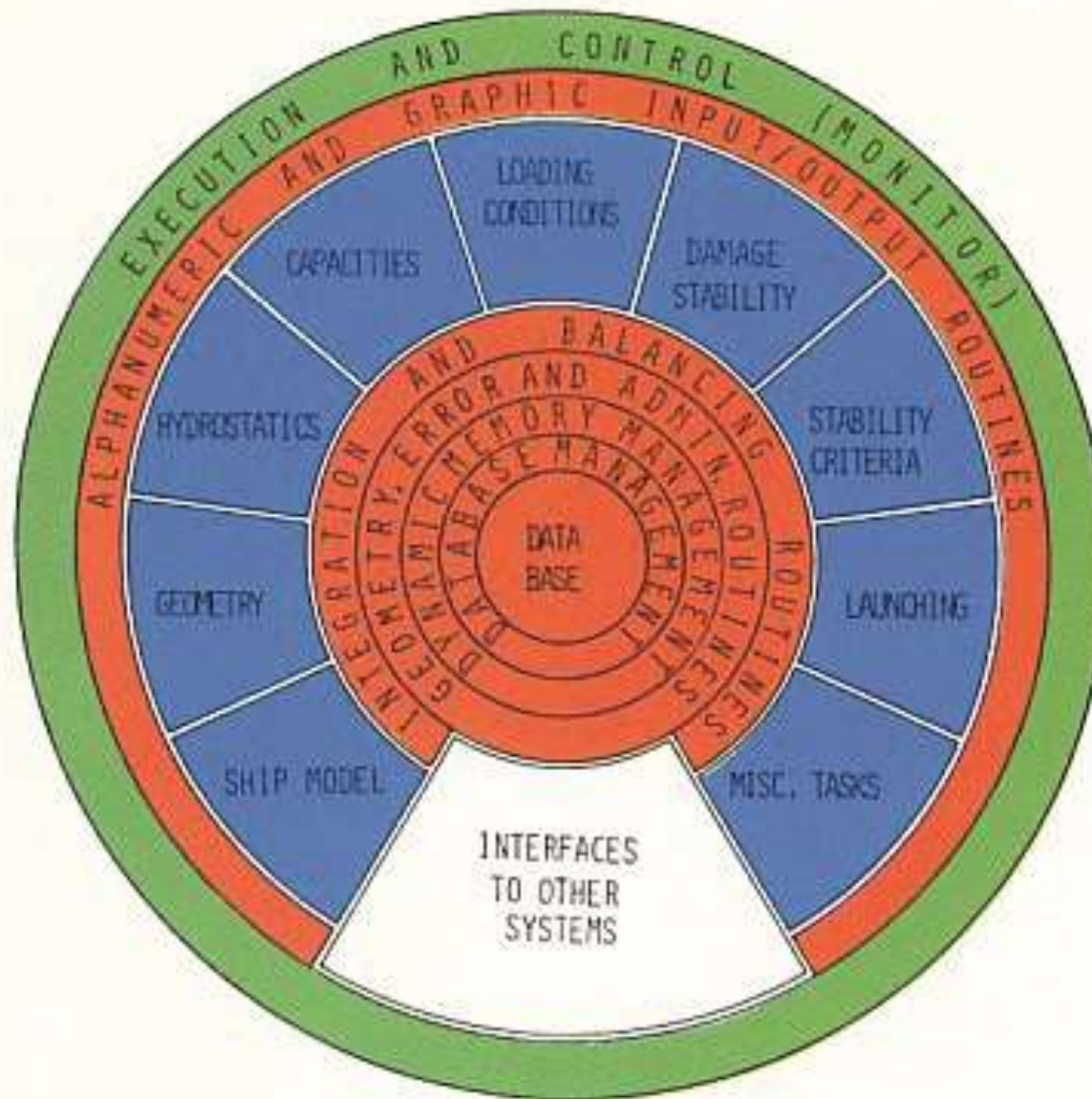
- NAPA for Design
 - For designing ships
 - 3D model
 - Ship technical calculations
- NAPA for Operations
 - For ship operation
 - Loading calculator
 - Voyage optimization
 - Electronic logbook
 - ...



NAPA for Design – a brief technical history

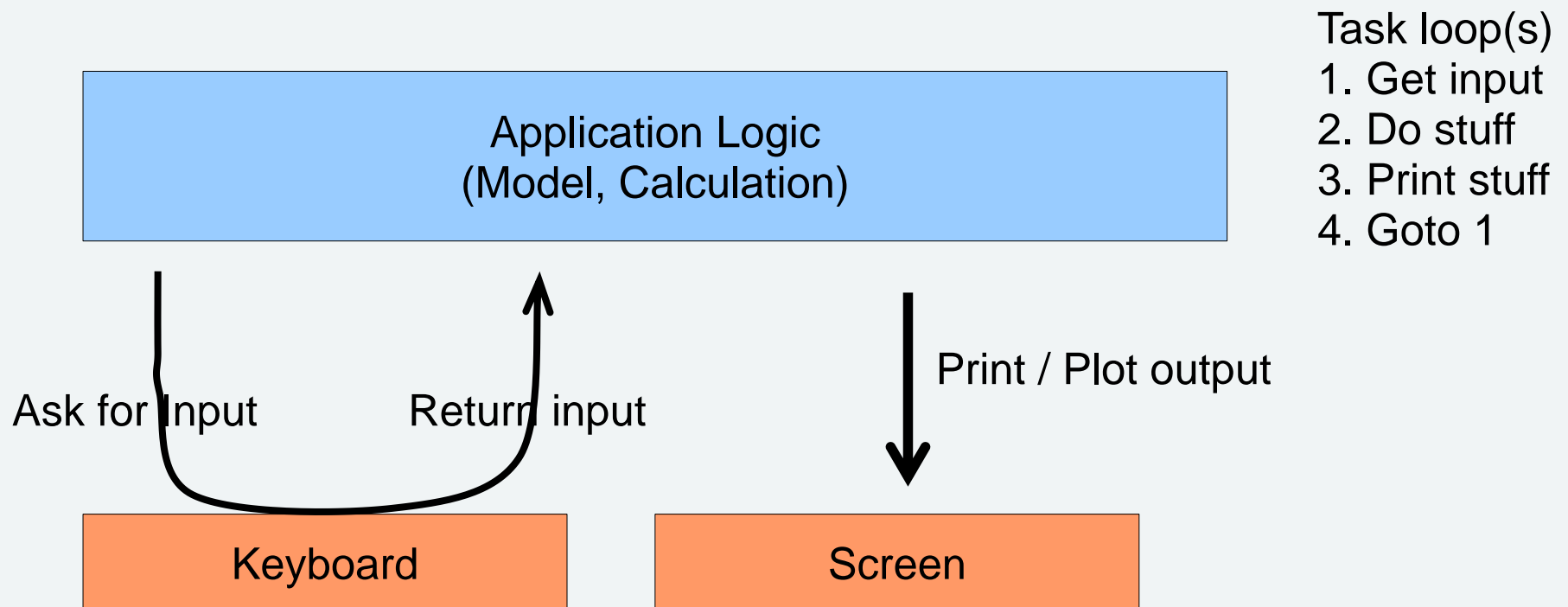
- Development started in late 1970s
 - Language: Fortran 77
- Oldest still present code from 1980
- Originally a couple of developers
 - Nowadays around 40-50 developers in Finland, Romania and India

NAPA architecture sketch from 1979



Dominant architectural metaphor

- NAPA is deep down a terminal application
 - Any place in code may ask for more input
- No layering, any subsystem may call any other (including asking for more input)



Situation before starting renewal

- ~2M lines of Fortran 77
 - A very primitive language
 - Newer Fortran (F90/95, F2003) taken into use only some years ago
- ~50000 lines of C (with a dash of C++)
- ~1.5M lines of Napa Basic
 - Mostly in the GUI
 - A primitive custom scripting language

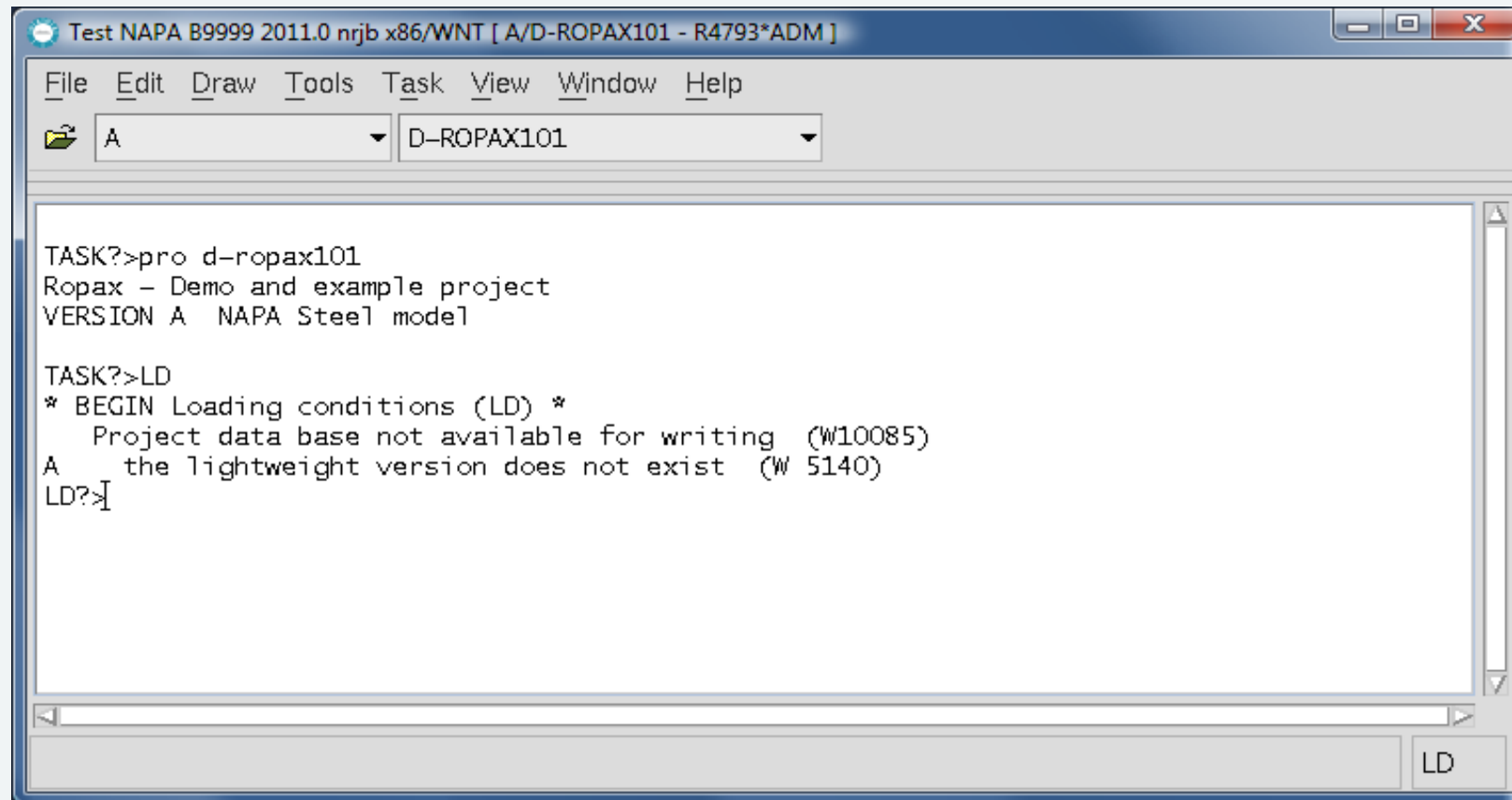
Situation before starting renewal

- State mostly in global variables
- Concept of "current object"
 - I.e. global variables representing e.g. a table or an image on which the routines operate
- Routine naming e.g. LD6541, CH17
 - Variable naming similar, e.g. CDBCCHL
- No named constants
 - E.g. reading RTYPE.EQ.3 you need to know that here 3 means string type
- ...

Situation before starting renewal

- >5% of all statements GOTOs
 - Not even counting Fortran's "alternate returns"
- Routines are
 - long (averaging 300-1000 sloc)
 - complex
 - highest cyclomatic complexity ~900
 - ~300 local variables not unheard of
- Etc, you get the picture

The session state in on the call stack



```
Test NAPA B9999 2011.0 nrjb x86/WNT [ A/D-ROPAX101 - R4793*ADM ]
File Edit Draw Tools Task View Window Help
A D-ROPAX101
TASK?>pro d-ropax101
Ropax - Demo and example project
VERSION A NAPA Steel model
TASK?>LD
* BEGIN Loading conditions (LD) *
Project data base not available for writing (W10085)
A the lightweight version does not exist (W 5140)
LD?>
```

1. Start Napa Motif GUI, Load Project, go to LD
2. Break to Debugger

The session state in on the call stack

Test NAPA B9999 2011.0 nrjb x86/WNT [A/D-ROPAX101 - R4793*ADM]

File Edit Draw Tools Task View Window Help

A D-ROPAX101

TASK?>pro d-ropax101

Ropax -
VERSION

TASK?>LD
* BEGIN
Proj
A the
LD?>

Call Stack

Name	Language
napaapi.dll!uiNextCommand(const char * prmpt, char * cmd, unsigned int ...)	C
napaapi.dll!NEXTCMD(const char * prmpt, unsigned int d1, const long * nc ...)	C
napaapi.dll!AI_TERMINAL_READ::CALL_NEXTCMD() Line 118 + 0x4a bytes	Fortran
napaapi.dll!AI_TERMINAL_READ(CHARACTER(76) PROMPT, .tmp..T6_V\$2 ...)	Fortran
napaapi.dll!AI_GET_INPUT_LINE(CHARACTER(76) PROMPT, .tmp..T98_V\$...)	Fortran
napaapi.dll!AI_GET_DATA_RECORD(CHARACTER(76) PROMPT, .tmp..T48_ ...)	Fortran
napaapi.dll!AI_READ_NEXT_COMMAND_A(PROMPT, .tmp..T68_V\$65, INT ...)	Fortran
napaapi.dll!LD01(INTEGER(4) NLEMS, CHARACTER(1) COMMND, .tmp..T ...)	Fortran
napaapi.dll!LDA() Line 44	Fortran
napaapi.dll!MN00(INTEGER(4) ... Line 418 + 0x5 bytes	Fortran
napaapi.dll!NAPA() Line 56	Fortran
napaapi.dll!startup(char ** argv, unsigned int argc, int mthr) Line 184	C
napaapi.dll!NapaMain(int argc, char ** argv, char ** envp) Line 65 + 0xf by	C

uiNextCommand: GUI Event Loop

LD01: LD Command Loop

MN00: Napa Top-Level Command Loop